WMGR: A Generic and Compact Routing Scheme for Data Center Networks

Daniela Aguirre-Guerrero, Miguel Camelo, Lluís Fàbrega, and Pere Vilà

***Abstract*— Data center networks (DCNs) connect hundreds and thousands of computers and, as a result of the exponential growth in their number of nodes, the design of scalable (compact) routing schemes plays a pivotal role in the optimal operation of the DCN. Traditional trends in the design of DCN architectures have led to solutions, where routing schemes and network topologies are interdependent, i.e., specialized routing schemes. Unlike these, we propose a routing scheme that is compact and generic, i.e., independent of the DCN topology, the word-metric-based greedy routing. In this scheme, each node is assigned to a coordinate (or label) in the word-metric space (WMS) of an algebraic group and then nodes forward packets to the closest neighbor to the destination in this WMS. We evaluate our scheme and compare it with other routing schemes in several topologies. We prove that the memory space requirements in nodes and the forwarding decision time grow sub-linearly (with respect to n, the number of nodes) in all of these topologies. The scheme finds the shortest paths in topologies based on Cayley graphs and trees (e.g. Fat tree), while in the rest of topologies, the length of any path is stretched by a factor that grows logarithmically (with respect to n). Moreover, the simulation results show that many of the paths remain far below this upper bound.**

***Index Terms*— Data center networks, compact routing, greedy routing, automatic groups, word-metric spaces.**

I. Introduction

DATA Center Networks (DCN) connect hundreds of thousands of computers to store, manage, and disseminate information. One of the major concerns in modern DCN is the exponential growth in their number of nodes. This growth not only has a negative impact on energy consumption, by increasing the operation cost, but also on data delivery because it reduces network performance. Therefore, designing scalable topologies and routing schemes to guarantee the optimal operation of DCN poses a serious challenge.

The problem known as Compact Routing (CR) consists of designing scalable routing schemes with respect to the number of network nodes n [1]. In CR schemes, the memory space requirements in nodes and the time to make forwarding decisions are scalable, i.e. grow sub-linearly with respect to n, while keeping the length of any path as close as possible to the shortest one. The path stretch is defined as the ratio between the length of a path found by the scheme and the corresponding shortest path.

In the routing scheme design, it is common to exploit some topological properties to establish workable CR schemes [2]. The topological properties that are desirable in DCN include: low hyperbolicity, low node degree, low diameter and multi-path [3], [4]. Hyperbolicity of a graph provides bounds on the distortion of the distances in the graph when it is embedded into an edge-weighted tree [3] and, for some geometric routing schemes, the stretch is proportional to the hyperbolicity. Low node degree results in servers and switches with a low number of ports; something which usually reduces cost. Low diameter (desirable logarithmic in terms of the number of nodes) reduces packet delay. Multi-path enables load balancing and fault-tolerant routing.

Routing schemes for DCN are usually designed to work on a particular topology, i.e. they are specialized schemes, and take advantage of the topological properties to become compact. Some examples of this are the BCube Routing Protocol (BRP) for the BCube topology [5], the DCell Fault-tolerant Routing protocol (DFR) for the DCell topology [6], the two-level routing scheme for the Fat-tree topology [7] and the routing schemes designed to work on Cayley Graphs (CG) [8]–[10]. In contrast to these specialized and compact routing schemes, generic routing schemes, i.e. topology independent, have also been proposed for DCNs. However, they are not compact. This is the case of the traditional routing scheme K-shortest path [11] that has been proposed to work on Jellyfish [12], Xpander [13] and Slim fly [14] topologies. This scheme can not be considered compact because of its high memory space requirements.

In this paper, we present the Word-Metric-based Greedy Routing (WMGR), a routing scheme that is compact and independent of the DCN topology. Our scheme extends and enhances the Greedy Routing (GR) scheme, introduced for the Internet in [15]. In the WMGR scheme, nodes are assigned to coordinates in the Word-Metric Space (WMS) of an algebraic group. Then, the information required to compute distances in the WMS is encoded using a Finite State Automaton (FSA). By proving that this FSA can be stored and processed in an efficient way, we guarantee that our scheme achieves scalable routing tables, node labels and forwarding decision time. Moreover, we prove that the WMGR scheme has low stretch in any DCN and stretch 1 in trees and topologies based on CG.

Manuscript received July 6, 2016; revised March 15, 2017, July 21, 2017, and September 28, 2017; accepted November 27, 2017; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor M. Andrews. Date of publication December 29, 2017; date of current version February 14, 2018. This work was supported in part by the Mexican Government under Grant CONACYT-SENER 2015-409697, in part by the Spanish Government under Grant TEC 2015-66412-R and Grant TEC 2015-71932-REDT, in part by the Catalan Government under Grant SGR-1469, and in part by the Flanders Government through the SMILE-IT Project—VLAIO. *(Corresponding author: Daniela Aguirre-Guerrero.)*

D. Aguirre-Guerrero, L. Fàbrega, and P. Vilà are with the Institute of Informatics and Applications, University of Girona, 17071 Girona, Spain (e-mail: daniela.aguirre@udg.edu; lluis.fabrega@udg.edu; pere.vila@udg.edu).

M. Camelo is with the Department of Mathematics and Computer Science, University of Antwerp, 2020 Antwerp, Belgium (e-mail: miguel.camelo@uantwerpen.be).

Digital Object Identifier 10.1109/TNET.2017.2779866

The remainder of the paper is as follows: Section II surveys the main proposals of routing schemes for DCN while Section III introduces the theoretical framework that supports our proposal. Section IV then presents the WMGR scheme and Section V introduces the performance evaluation of specialized CR schemes and the WMGR scheme working on different DCN topologies. Finally, Section VI provides the conclusions.

II. Related Work

In this section, we review the main routing schemes and topologies that have been proposed for Data Center Networks (DCN). We describe their characteristics, whether they are compact or not, and whether they are specialized (topology dependent) or generic (topology independent) in DCN.

The BCube Routing Protocol (BRP) for BCube Topology [5]

A BCube topology is defined by two parameters p and q and uses the $q-$dimensional Hypercube as the underlying interconnection network. Each of the q levels in BCube consists of p switches connected to each others. Each switch in level 0 is connected to q servers. The forwarding algorithms of BPR use a divide-and-conquer approach to route data packets. This is possible because BRP performs hierarchical routing based on BCube levels.

The DCell Fault-Tolerant Routing Protocol (DFR) for DCell Topology [6]

Similar to the BCube, the DCell topology is defined by parameters p and q. This topology is recursively defined through q levels. The Level 0 is given by a set of blocks of p servers connected to a switch. The higher-levels of the topology are constructed by putting together blocks from the lower-levels. The forwarding algorithm of DFR finds the next node in the path in a recursive way. It assumes that source and destination nodes are on different and faraway levels. This approach may be not suitable in terms of time when the source and destination nodes are on the same topology level.

The Two-Level Routing Scheme for Fat-Tree Topology [7]

The Fat-tree topology is a kind of Clos topology. This topology is defined by the parameter r which is an even number that determines the number of ports in the switches, meanwhile the servers have only one port. Fat-tree is a hierarchical topology with 3 levels of switches and a constant diameter with a value 6. The two-level routing scheme is hierarchical and applies a centralized mechanism to create the routing tables that provide path redundancy.

Routing Schemes Based on Permutation-Sort for Cayley Graph (CG)

Nodes in a CG that arise from permutation groups can be represented by the set of permutations of the array $[0, \ldots, s-1]$ and are connected by a permutation rule [16]. These topologies have a regular degree whose value depends on the parameter s, except the Butterfly graph that is a 4-regular graph. The majority of routing schemes proposed for topologies with underlying CG are based on permutation-sort. We can mention those designed for the topologies based on Hypercube, Butterfly [8], Star [9], and Pancake [10] graphs. The forwarding algorithms based on permutation-sort follow a Greedy Routing (GR) strategy, where nodes forward packets to the nearest adjacent node according to the distance computed in terms of the number of permutations. Because of the shortest path problem being equivalent to finding an optimal sorting of an array, these routing schemes have stretch 1.

Routing Scheme for Any CG [17]

In this work, the authors prove that CG can be represented by generalized chordal rings. Then, they propose a routing scheme for chordal rings that is based on look-up tables. The memory space requirements per node is $O(n)$ bits (which is equivalent to having full routing tables) and the time complexity is $O(D)$ time units, where n and D are the number of nodes and the diameter of the network, respectively.

The K-Shortest Path Routing Scheme [11] for Jellyfish [12], Xpander [13] and Slim fly [14] Topologies

These topologies are based on graphs related to the degree-diameter problem, i.e. finding the largest regular graph for a given degree and diameter [18]. Nodes in the regular graph represent Top-of-Rack switches that are connected to the same number of servers. In these topologies, the authors propose using the K-shortest path routing. As this routing scheme uses precomputed full routing tables, it is able to provide enough path diversity to exploit the network's capacity. Then, the memory space requirements per node is $O(n)$ bits.

Table I summarizes the properties of the topologies on which the discussed routing schemes work. It is important to note that the values presented in Table I may differ from those introduced in the original papers, as some authors do not take into account the number of servers to compute n, D and Δ, whereas we do take into account servers and switches as part of the network topology. As we can see, the values of D and Δ are sub-linear, and in some cases constant, with respect to n. In Section V-B, we prove that the schemes BRP, DFR, Two-level routing and routing based on permutation-sort exploit the sub-lineal values of D and Δ to become compact. However, they are specialized. On the other hand, the routing scheme for any CG and K-shortest path routing scheme are generic but they are not compact due to their high memory space requirements.

To the best of our knowledge, there is no generic and CR scheme for DCN. To overcome this limitation, we present the Word-Metric-based Greedy Routing (WMGR), which is compact and independent of network topology. Our scheme follows a GR strategy, where nodes are assigned to coordinates (labels) of a metric space, a process called embedding. Then, nodes forward data packets to their adjacent node that is the nearest to the destination in the metric space.In our scheme, graphs are embedded into Word-Metric Spaces while much of the research works in GR focus on the Euclidean [19], [20] and Hyperbolic [21], [22] spaces. These works, however, have some of the following problems: 1) they do not guarantee packet delivery, or 2) they are not scalable in space and/or

TABLE I
DATA CENTER NETWORKS AND THEIR TOPOLOGICAL PROPERTIES

Topology / Configuration parameters[a]	Total of nodes (n)	Diameter (D)	$O(D)$	Maximum node degree (Δ)	$O(\Delta)$
BCube / p, q	$p^q(q+p+1)$	$p+3$	$O(n^{1/4})$	p	$O(n^{1/4})$
DCell / p, q	$\left(\left(\left(p+\frac{1}{2}\right)^{2q}-\frac{1}{2}\right)\left(1+\frac{1}{p}\right),\left((p+1)^{2q}-1\right)\left(1+\frac{1}{p}\right)\right)$	$\leq 2^{q+1}+3$	$O(1)$		$O(n^{1/6})$
Fat-tree / r	$\frac{r^2(r+5)}{4}$	6		r	$O(n^{1/3})$
Jellyfish / p, q, r	$q(p-r+1)$	$\leq \lceil \log(q) \rceil + 3$	$O(\log(n))$	p	$O(n^{1/c})$, for $1 < c \leq D$
Xpander / p, q	$qp\left(\lceil \frac{p}{2} \rceil + 1\right)$			$\lceil \frac{3p}{2} \rceil$	
Slim fly / p, q	$q\left(\lceil \frac{p}{2} \rceil + 1\right)$	4	$O(1)$		$O(n^{1/2})$
Hypercube / s	2^s	s	$O(\log(n))$	s	$O(\log(n))$
Bubble-sort / s	$s!$	$\frac{s^2-s}{2}$	$O(\log(n)^2)$	$s-1$	
Star / s		$\lfloor \frac{3(s-1)}{2} \rfloor$	$O(\log(n))$		
Transposition / s		$s-1$		$\frac{s^2-s}{2}$	$O(\log(n)^2)$
Butterfly / s	$2^s s$	$\lfloor \frac{3s}{2} \rfloor$		4	$O(1)$

time (see [15, Sec. I]). Another example of GR, specialized for wireless sensor networks, is [23].

III. PRELIMINARIES

In this section, we introduce some notation, terminology and concepts that are needed to understand and support our proposal. We assume that the reader has some basic knowledge of metric spaces and group theory, otherwise we refer the reader to [24] and [25].

A. Greedy Embedding

A graph embedding (or network embedding) is the process of assigning coordinates (labels) in a metric space (X, d_X) to the set of nodes (vertices) from a connected graph Γ, where X is a set (of coordinates) and d_X is a distance function over X (see [24, Definition 1.1]). Then, network embedding is given by an injective map

$$\pi : V(\Gamma) \rightarrow X, \tag{1}$$

where $V(\Gamma)$ is the set of nodes of Γ.

Taking advantage of those labels, a Greedy Routing (GR) strategy can be performed in order to route data packets between any source and destination nodes. This strategy starts at the source node by executing a forwarding algorithm that computes the distances to its adjacent nodes and forwards data packets to the nearest node to the destination in the metric space. Then, the same forwarding algorithm is executed in each previously selected node until data packets reach the destination node.

To ensure that the GR strategy explained above always succeeds in finding a path between any two nodes (and therefore packet delivery is guaranteed), the network embedding (1) must be *greedy* [26].

Definition 1: Let π be a network embedding given by (1). We say that π is *greedy* if it satisfies that

$$\forall y, z \in V(\Gamma), \quad \exists i |\ d_X(\pi(i), \pi(z)) < d_X(\pi(y), \pi(z)), \tag{2}$$

where y and i are adjacent nodes, and y is different to z.

Condition (2) ensures that for any two distinct nodes y and z, node y is able to find among its adjacent nodes, a node i that is closer (according to d_X) than itself to node z.

Given a connected graph Γ, an interesting metric space is the one obtained by defining $V(\Gamma)$ as the set of elements, and defining the distance between two nodes, i.e. d_Γ, as the number of edges of the shortest path connecting them. This metric space is called the *metric graph*. Note that if (1) preserves the distances of the metric graph, then the GR strategy routes data packets along the shortest paths in the network. The network embedding that preserves distances is called *isometry*.

Definition 2: Let π be a network embedding given by (1). We say that π is an *isometry* if it satisfies that

$$\forall y, z \in V(\Gamma), \quad d_X(\pi(y), \pi(z)) = d_\Gamma(y, z), \tag{3}$$

B. Cayley Graphs and Word-Metric Spaces

In this subsection, we show that Cayley Graphs (CG) can be represented as a metric space called Word-Metric Space (WMS) which is isometric to the metric graph of the CG. For more details about CG and WMS, we refer the reader to [27].

Definition 3: Let $G =< S|R >$ be a finitely presented group, where S and R are the set of generators and relators, respectively (see [25, Sec. 2.2]). The graph $\Gamma(G, S)$ is called

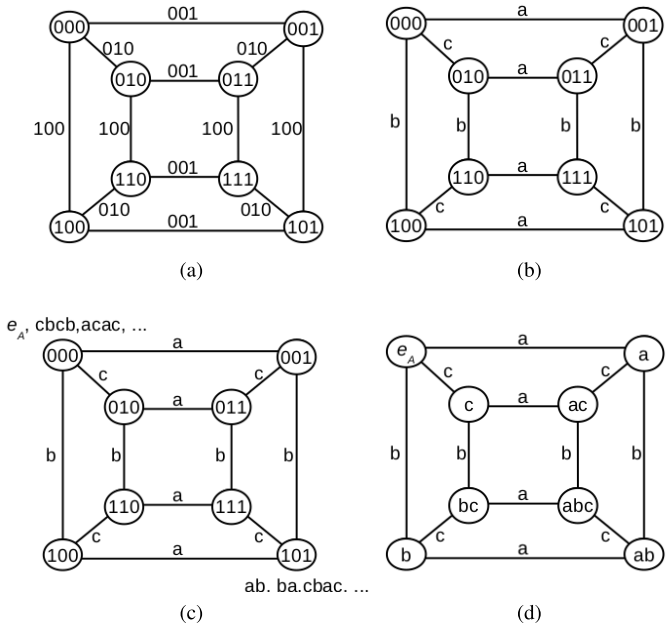

Fig. 1. **(a)** Cayley graph of group $(G, \oplus)$ with respect to the generating set S, where $G = \{000, 001, 010, 011, 100, 101, 110, 111\}$, $\oplus$ is the symbol for the XOR operation and $S = S^{-1} = \{001, 010, 100\}$. **(b)** Map (4) assigns letters in an alphabet, e.g. $A = \{a, b, c\}$, to each generator (edge), e.g. $\phi(a) = 001$, $\phi(b) = 100$ and $\phi(c) = 010$. **(c)** Map (5) assigns a set of words in $F(A)$ to each node. These words represent paths from e_A to said node, e.g. $\pi^{-1}(000) = \{e_A, cbcb, acac, \ldots\}$. **(d)** Given a lexicographical order over A, e.g. $a < b < c$, the language L is defined by (6). Map (7) assigns a unique word in L to each node.

the *Cayley Graph* (CG) of G with respect to S, if each group element in G corresponds to a node in $V(\Gamma(G, S))$, i.e. there is a bijective map $\phi : G \rightarrow V(\Gamma(G, S))$, and two nodes $y, z \in V(\Gamma(G, S))$ (where $y = \phi(g)$ and $z = \phi(h)$) are adjacent, if g and h are elements of G that satisfy $g \cdot s = h$ for some $s \in \{S \cup S^{-1}\}$, where S^{-1} is the set of inverses of G (see Fig. 1a).

In the remainder of this paper, group elements $g \in G$ and nodes $\phi(g) \in V(\Gamma(G, S))$ are used interchangeably.

Definition 4: Let $F(S) =< S|\emptyset >$ be the finitely presented group that does not have relators, $F(S)$ is then called the *Free Group* (FG). Let $\Gamma(F(S), S)$ be the CG of $F(S)$, then $\Gamma(F(S), S)$ is a $2|S|$-regular tree (see [28, Th. 3.20]). Since the FG does not have a set of defined relators, its cardinality is infinite, and consequently its CG is also infinite (see Fig. 2) in accordance with Definition 3.

Definition 5: Let A be a finite alphabet (i.e. a set of symbols). We define an *alphabet closed under inversion* given by $\{A \cup A^{-1}\}$, where $|A| = |A^{-1}|$ and A^{-1} is an alphabet disjoint from A. Then, there is an involution $i : A \rightarrow A^{-1}$ and $i(a) \in A^{-1}$ is called the *inverse* of $a \in A$. Henceforth, we write a^{-1} instead of $i(a)$ and the term *alphabet* will be used to refer to an alphabet close under inversion.

Definition 6: Let A be an alphabet, a *word over* A is given by a string of symbols over $\{A \cup A^{-1}\}$. The *inverse of a word* w, denoted by w^{-1}, is the reverse string given by the symbols inverses of w. A *reduced word* is a word that does not have substrings of the form ww^{-1}.

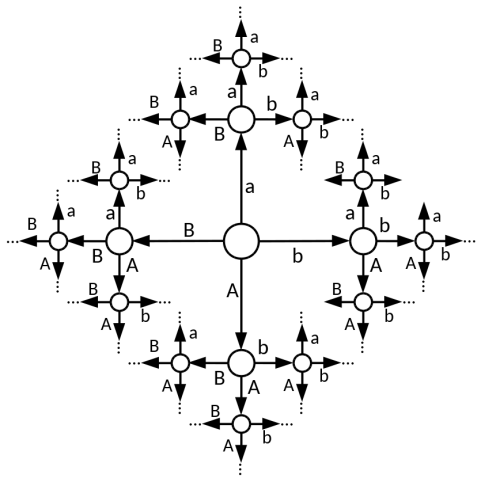

Fig. 2. Part of the Cayley graph of the free group with $S = \{a, b\}$ and $S^{-1} = \{A, B\}$ as generating and inverse sets, respectively.

Definition 7: Let A be an alphabet. We define the set of all words over A denoted by A^*, which includes the *null string* denoted by e_A, where $|e_A| = 0$.

Definition 8: Let A be an alphabet. The *Free Group over* A denoted by $F(A)$ is given by the set of all reduced words over A (including e_A). The *group operation* is the string concatenation where substrings of the form ww^{-1} are cancelled. The *identity element* is e_A.

Definition 9: Let $G =< S|R >$ and $\Gamma(G, S)$ be a finitely presented group and its CG, respectively. Let A be an alphabet such that $|A| = |S \cup S^{-1}|$. We define a biyective map

$$\phi : A \rightarrow \{S \cup S^{-1}\}, \tag{4}$$

and a *surjective group homomorphism* (see [25, Sec. 1.4]) given by the map

$$\pi : F(A) \rightarrow G, \tag{5}$$

where $\pi(w)$ is the element $g \in G$ represented by $w \in F(A)$ under π (see [27, Sec. 2.1]). Since each group element can be represented by one or more words in $F(A)$, we can define an *equivalence relation*, denoted by $=_G$, such that $w =_G v$, if w and v represents the same group element, i.e. $\pi(w) = \pi(v)$. Then, the set of words associated to a group element is given by an *equivalence class* denoted by $[w]$.

From Definition 3, the set of edges incident to a node in $\Gamma(G, S)$ is given by the set of generators and inverses, i.e. $\{S \cup S^{-1}\}$. Therefore, for each node in $\Gamma(G, S)$, (4) assigns to each of its incident edge a letter in A (see Fig. 1b). Note that words in $F(A)$ can be seen as paths in $\Gamma(G, S)$. On the other hand, homomorphism (5) assigns one or more words in $F(A)$ to each node in $\Gamma(G, S)$ as follows: e_A is assigned to the node that represents the identity element of G. Then, a set of words $[w]$ is assigned to a node g, if words in $[w]$ represent paths in $\Gamma(G, S)$ that go from the node e_A to the node g (see Fig. 1c and [27, Sec. 2.2]).

In order to assign a unique word in $F(A)$ to each node in $\Gamma(G, S)$, a canonical form of the set of equivalence classes $[w]$ needs to be defined.

Definition 10: Let $<_A$ be a *lexicographical* order over the alphabet A. Let u and v be two words in A^*. We say

that u is *ShortLex* than v, if u is shorter than v, i.e. $|u| < |v|$, or u and v have the same length and u comes before v in the order $<_A$.

Definition 11: Let $L \subseteq F(A)$ be a language (i.e. a set of words) such that

$$L = \{u \in F(A) \mid \forall v : \ v =_G u \text{ and } u <_A v\}, \tag{6}$$

Then the map

$$\pi : L \to G \tag{7}$$

is an isomorphism and words in L represent a set of shortest paths in $\Gamma(G, S)$. Moreover, since L gives a canonical form of the set of equivalence classes $[w]$, then (7) assigns a unique word in L to each node in $\Gamma(G, S)$ (see Fig. 1d).

From the previous discussion, we can define the WMS related to a CG, which allows the distance between two nodes in the CG to be measured.

Definition 12: Let $G =< S|R >$ be a finitely presented group and let L be a language over an alphabet A, such that G and L satisfy the isomorphism defined by (7). The *Word-Metric Space* (WMS) of $G =< S|R >$ is given by the set A^* and the Word-Metric (WM)

$$d_A(u, v) = |w|, \quad \textit{where } \pi(w) =_G \pi(u^{-1}v) \textit{ and } w \in L. \tag{8}$$

The WM measures the length of the shortest path between any two nodes $s = \pi(u)$ and $d = \pi(v)$ in $\Gamma(G, S)$. Therefore, the WMS of $G =< S|R >$ preserves the distances in the metric graph $\Gamma(G, S)$.

From Definition 6, the word $u \in L$ represents one of the shortest paths in $\Gamma(G, S)$, which goes from the node e_A to the node $g = \pi(u)$, then u^{-1} represents a path from g to e_A. Moreover, the word $u^{-1}v$, where $h = \pi(v)$, describes a path between nodes g and h. This path goes from g to e_A and from e_A to h. Therefore, finding the shortest path between any pair of nodes $g, h \in \Gamma(G, S)$ is equivalent to finding the canonical form of word $u^{-1}v$, i.e. a word $w \in L$ such $\pi(w) =_G \pi(u^{-1}v)$. This problem is called the Minimum Word Problem (MWP) [29] and can be resolved by applying a set of rewriting rules (related to the group presentation) over the word $u^{-1}v$ (see Fig. 3).

The following section explains how to embed a network into the WMS of a finitely presented group and how compute the distances between the nodes in the network by using (8).

IV. Word-Metric-Based Greedy Routing Scheme

In this section, we present the Word-Metric-based Greedy Routing (WMGR) scheme, which is independent of the Data Center Network (DCN) topology. The WMGR scheme performs a network embedding into the Word-Metric Space (WMS) of an algebraic group. Then, the packet forwarding follows a Greedy Routing (GR) strategy.

A. Embedding Graphs Into Word-Metric Spaces

Definition 13: Let Γ be a connected and finite graph representing a network topology whose maximum node degree is denoted by Δ. We define the following network embedding

$$\pi : V(\Gamma) \to L, \tag{9}$$

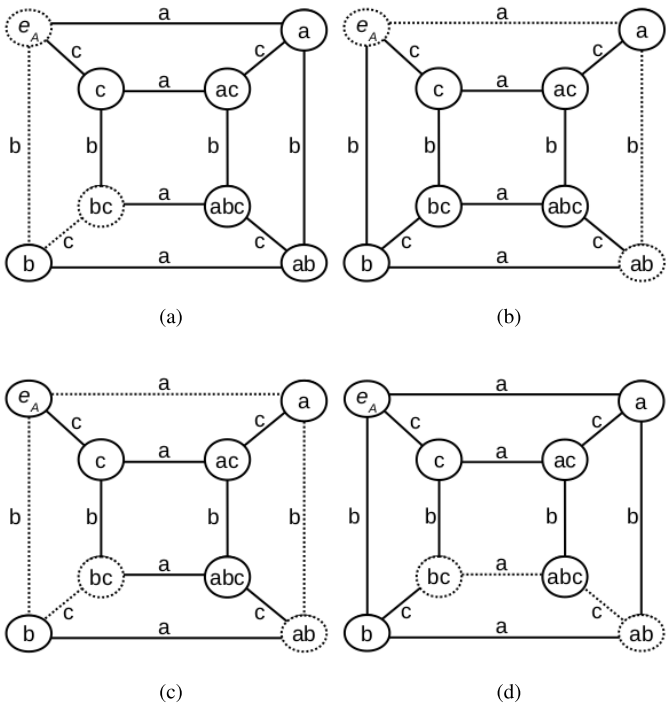

Fig. 3. **(a)** bc represents one of the shortest paths from e_A to bc, whereas $(bc)^{-1} = cb$ represents one of the shortest paths from bc to e_A. **(b)** ab represents a shortest path from e_A to ab. **(c)** $(bc)^{-1}ab = cbab$ represents a path from bc to ab. **(d)** By applying a set of rewriting rules over $cbab$, we can obtain the word $ac \in L$ which represents one of the shortest paths from bc to ab.

where $V(\Gamma)$ is the set of nodes (vertex) of Γ and L is a language over an alphabet A, such that

$$|A| = \Delta \tag{10}$$

and L satisfies Definition 12 where:

1) **For topologies with an underlying CG**, i.e. $\Gamma = \Gamma(G, S)$, L corresponds to the WMS of $G =< S, R >$.
2) **For any topology**, L corresponds to the WMS the Free Group (FG), i.e. $F(A) =< A|\emptyset >$.[1]

In Section V-F, we show that if a network is embedded into the WMS of its underlying CG, the WMGR scheme provides stretch 1 but it may increase the memory space requirements per node. We also show that if the network is embedded into the WMS of the FG, the WMGR scheme provides logarithmic stretch and has low memory space requirements per node.

The first step in the embedding process is to define an alphabet A that satisfies (10) and a lexicographical order $<_A$ over this alphabet. Then, Algorithm 1[2] presents the steps to carry out the network embedding introduced in Definition 13. In steps 1 and 2 of Algorithm 1, nodes are enumerated by computing a Breadth-First Search (BFS) tree; in step 3, the

[1]Notice that a CG can also be embedded into the WMS of the FG.

[2]When the network is embedded into the WMS of its underlying CG, Algorithm 1 assumes that nodes know which generator is associated to each of their incident links (see Fig. 1b and 4a). Otherwise, it would be possible to perform an exploration technique such as BFS or Depth-First Search (DFS), to discover the path associated to the set of relators and derive the corresponding generator of each link.

Algorithm 1 Network Embedding Into the Word-Metric Space

Input: An alphabet A that is in lexicographical order $<_A$.

1: Select an arbitrary node r of $V(\Gamma)$.
2: Compute a breadth-first search tree T_Γ rooted at r.
{If the word-metric space corresponds to the underlying Cayley graph, nodes must be visited according to $<_A$.
If the word-metric space corresponds to the free group, when a node is visited, it must identify its links with letters in A in the order given by $<_A$.}
3: Labelling r with a special symbol $e_A \notin A$.
4: **for** each node v in $V(T_\Gamma)$ **do**
5: **for** each child u of node v **do**
6: labelling u with the resulting word from the concatenation of the v's label and the letter $\phi^{-1}(s)$, where s is the generator corresponding with the link (u, v). {when $v = r$, symbol e_A is omitted from label of u}
7: **end for**
8: **end for**

root of the BFS tree is labelled with a special symbol, i.e. e_A, that denotes the null string. The *for*-loops (lines 4 and 5) explore the network (according to the previous enumeration) in order to assign the node labels. When a node has its turn to be labelled (step 6), its label is built by concatenating the label of its parent (in the BFS tree) with the corresponding letter in A that connects it with its parent (see Fig. 4c and 5b).

In the following section, we show that the set of node labels, i.e. L, can be obtained from a set of Finite States Automata (FSA) related to the WMS. However, Algorithm 1 builds the node labels during the embedding process in order to support a distributed configuration of the WMGR scheme. In fact, all the steps in Algorithm 1 can be performed in a distributed way, for instance, step 2 can be implemented by using the distributed BFS algorithm presented in [30].

B. Distance Computation Using Finite State Automata

From Definition 12, computing the distance (in a WMS) between two nodes labeled as u and v involves: 1) finding the canonical form of the word $u^{-1}v$, i.e. to solve the Minimum Word problem (MWP) and 2) computing the size of the resulting word. The canonical form of $u^{-1}v$ can be obtained in $O(|u^{-1}v|^2)$ time units by applying a set of rewriting rules over $u^{-1}v$ (see [27, Th. 2.3.10]). This set of rewriting rules is associated to a group presentation (and therefore to a CG).

ShortLex automatic groups are algebraic groups whose set of rewriting rules can be encoded in a *ShortLex* Automatic Structure (SAS). Let $G =< S|R >$ be a *ShortLex* automatic group, then the SAS of $G =< S|R >$ consists of:

- A *ShortLex* order $<_A$ over an alphabet A, such that A and S satisfy (4).
- An FSA called *word acceptor* (*WA*). This automaton accepts the language defined by (6).

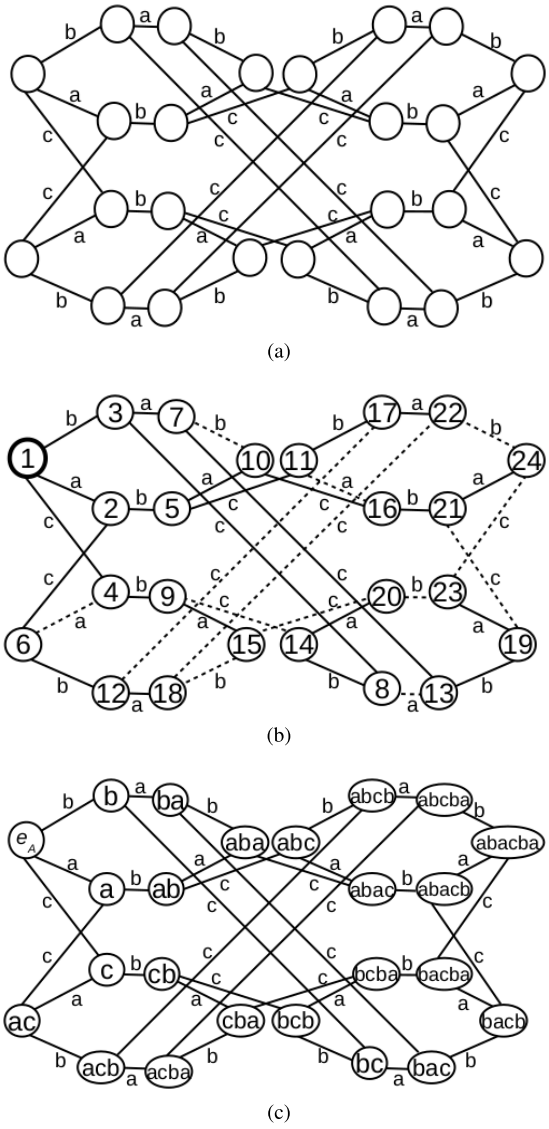

Fig. 4. Embedding the Cayley graph called Bubble-sort [10] into its own word-metric space. (a) Links are identified with the letter in the alphabet $A = \{b,, a, c\}$ that corresponds to their group generator according to (4). (b) Nodes are enumerated in breadth-first search order and following the lexicographical order $a < b < c$. (c) Label assignment.

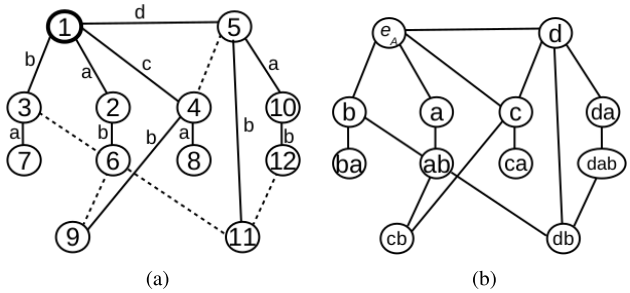

Fig. 5. Embedding a connected graph into the word-metric space of the free group. (a) Nodes are enumerated in breadth-first search order and links are identified with a letter in A following the lexicographical order $a < b < c < d$. (b) Label assignment.

- An FSA over (A, A), for each $a \in A \cup \{e_A\}$, called *multiplier automaton* (M_a), that accepts two words if, and only if, they represents paths in $\Gamma(G, S)$ that start in the same node and whose end nodes are adjacent.

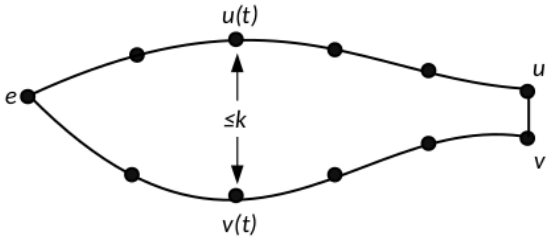

Fig. 6. Geometric representation of the *fellow-traveller* property. Distances between end nodes of paths $\overline{eu}(t)$ and $\overline{ev}(t)$ are bounded by an integer k.

An important feature of *ShortLex* automatic groups is that their associated CG has the *fellow-traveller* property.

Definition 14: Let M_a be a multiplier automaton of a *ShortLex* automatic group $G =< S|R >$. Let u and v be words such that $(u, v) \in M_a$. If the distance in the WMS (see (8)) between prefixes of u and v with same length t, i.e. $d_A(u(t), v(t))$, is bounded by an integer k, then it is said that $\Gamma(G, S)$ has the *fellow-traveller* property.

From a geometric point of view, if $\Gamma(G, S)$ has the *fellow-traveller* property, then the distance between every pair of paths in $\Gamma(G, S)$ that begin in the same node and whose last nodes are adjacent is bounded by an integer k (see Fig. 6). It implies that paths associated with a pair of words $(u, v) \in M_a$ have an associated set of *word differences* whose length is, at most, k.

Definition 15: Let M_a be a multiplier automaton of a *ShortLex* automatic group $G =< S|R >$. We define the set of *word differences* of $G =< S|R >$, denoted by *WD*, as the union of *word differences* associated with all pairs of words that satisfy the Definition 14, i.e.

$$WD = \{w \in L(WA) : |w| \leq k\}, \tag{11}$$

The set *WD* represents a set of nodes (words) in the subgraph given by the ball of radio k around to the identity element, denoted by $B(Id, k)$. This set is of major importance in the computation of an SAS, as it encodes the information of the topological structure of $\Gamma(G, S)$ [31].

We are interested in finding structures that are able to solve the MWP because this will also allow us to determine the distance between nodes in the WMS generated by the CG. Although the MWP can be solved efficiently in time by using the *WA* and the M_a, there are alternative structures that are created during the process of building the SAS and can also solve the MWP. The knuth-Bendix procedure [32] provides an efficient method to find an SAS from a group presentation. In our proposal, the alphabet associated to the SAS is the same as that used during the embedding process. The intermediate structures that are created during the execution of the knuth-Bendix procedure and can also solve the MWP are the following:

1) Confluent Rewriting System (CRS): A *CRS* is a system of rewriting rules which consists of equations in the form of $w_1 \rightarrow w_2$, where $w_1 =_G w_2$ for all $w_2 \in WD$ and w_2 is the canonical form of w_1. In this way, any word that is not in canonical form is composed by subwords w_1 that are in the left-hand side of the rules. Then, any word in A^* should be reduced in a finite number of steps by replacing its substrings w_1 with their corresponding right-hand side w_2 in the *CRS*.

2) Index Automaton (IA): An *IA* identifies subwords in any word that is not in canonical form with left-hand side in the *CRS*. In fact, for solving the MWP, *IA* is faster than *CRS*. This is because *IA* could be used to replace those subwords with their corresponding right-hand side in the *CRS*.

3) Word Differences Automaton (Diff): This automaton accepts (u, v) for all $u, v \in A^*$ if, and only if, the canonical form of $u^{-1}v$ is $w \in$ *WD*, then w is the final state. The set of states of *Diff*, denoted by σ_{Diff}, is given by the set of *WD*, therefore $\sigma_{Diff} \subseteq V(B(Id, k))$.

The following structures are able to solve the MWP:

- *WA + Diff*
- *CRS + IA*
- *CRS*
- *Diff*

The results from experimental evaluation of the size of these structures are presented in [33]. These results show that the *Diff* has the lowest memory space complexity between the above-mentioned structures, thanks to the fact that *Diff* encodes the equations of a *CRS*. In fact, although the number of equations in the *CRS* may be infinite, the same information can also be encoded in a *Diff* (see [27, Lemma 6.3.4]). In Section V-B, we analyse the impact of the size of *Diff* on the memory space complexity per node of the WMGR scheme. We will show that the size of *Diff* related to the FG is $O(|A|)$ bits. Additionally, we will discuss the relationship between the size of *Diff* and the hyperbolicity of five well-known families of CG.

C. Greedy Forwarding Algorithms

The procedure for forwarding data packets between two nodes starts at the source node by computing the distances from its adjacent nodes to the destination node. Then, data packets are forwarded to the nearest node to the destination node. The same procedure is repeated in the previously selected node until data packets reach the destination node.

The forwarding process in topologies embedded into the WMS of the FG is performed by using Algorithm 2. Note that packets can be forwarded through links that are not in the spanning tree, because nodes find among all their adjacent nodes the nearest one to the destination. For example, in Fig. 5, when routing packets from node *cb* to node *a*, node *cb* computes the distance in the WMS from *ab* to *a*, which is 1, and from *c* to *a*, which is 2. Then packets are forwarded to *ab*, although edge *(cb,ab)* is not in the spanning tree (see Fig. 5a).

The forwarding process in topologies embedded into the WMS of their underlying CG is performed by using Algorithm 3, which improves the temporal complexity with respect to Algorithm 2 and the forwarding technique proposed in [15] and [34]. Let u and v be node labels of a source and a destination nodes. If the word w is the canonical form of $u^{-1}v$, then w represents the sequence of edges in the shortest path between the source and the destination node. Thus, we can select the nearest node to a destination node as the one that is connected by the edge (generator) given by the first letter in w.

Algorithm 2 Greedy Forwarding in Topologies Embedded Into the Word-Metric Space of the Free Group

Input: A list l of labels of the adjacent nodes. Label of the destination node (v).

Set *distance* $\leftarrow \infty$
for u in l **do**
  Set $w \leftarrow u^{-1}v$
  Set $w_{red} \leftarrow$ *reduce*(w)
  if *distance* $> |w_{red}|$ **then**
    Set *next_node* $\leftarrow u$
    Set *distance* $\leftarrow |w_{red}|$
  end if
end for
Forward data packets to *next_node*

Algorithm 3 Greedy Forwarding in Topologies Embedded Into the Word-Metric Space of Its Underlying Cayley Graph

Input: Label of the node that is executing the forwarding procedure (u). Label of the destination node (v)

Set $w \leftarrow u^{-1}v$
Set $w_{red} \leftarrow$ *reduce*(w)
Set $e \leftarrow$ the first letter in w_{red}
Forward data packets to the node connected to edge e

This procedure works because, as we prove in Section V-F, our network embedding is isometric for topologies based on trees or embedded into its own CG, i.e. words in canonical form are congruent with shortest paths in the network.

The core of algorithms 2 and 3 is the *reduce* procedure, which computes the canonical form of any word by solving the MWP. Although there are many equivalent ways to solve the MWP by using *Diff* (see [35, Sec. 13.3.2], [27, Sec. 6.3]), we propose Algorithm 4. From the definition of *Diff* (see Section IV-B3), if *Diff* accepts (e_A, u), then the final state u_{red} is the canonical form of word u. Therefore, any word $w \in A^*$ can be reduced in a finite number of steps by replacing its substring u with u_{red}.

Algorithm 4 Compute the Canonical Form of a Word

Input: The *Diff* automaton. A word $w \in A^*$

Set $w_{red} \leftarrow w$
while w_{red} has substrings u such (e_A, u) is accepted by *Diff* **do**
  for each u **do**
    Set $u_{red} \leftarrow$ final state of *Diff* when read (u, e_A).
    Replace u with u_{red} in w_{red}
  end for
end while
return w_{red}

V. Performance Evaluation

In this section, we evaluate our scheme and compared it to other routing schemes in different topologies presented in Section II: the specialized routing schemes BCube Routing Protocol (BRP), DCell Fault-tolerant Routing protocol (DFR), two-level routing scheme and the routing schemes based on permutation-sort; the topologies BCube, DCell, Fat-tree, Jellyfish, Xpander, Slim fly, and five Cayley Graphs (CG), Hypercube, Bubble-sort, Star, Transposition and Butterfly.

A. Performance Metrics

We use traditional performance metrics in Compact Routing (CR) [1]:

1) **Memory space requirements**. The amount of memory in bits used by a node to store its label, routing table and forwarding algorithm.
2) **Forwarding decision time**. The number of steps that the forwarding algorithm takes to find the next node in the path.
3) **Stretch**. The ratio between the length of the path found by the scheme and the corresponding shortest path between any pair of nodes in the network. The stretch of a scheme is the highest stretch among all source-destination pairs.

In the following subsections, we present a complexity analysis of these performance metrics for the above-mentioned specialized CR schemes and for the WMGR scheme. Additionally, through computer simulations, we evaluate the memory space requirements of the WMGR scheme and the stretch of those schemes that do not compute the shortest paths.

B. Memory Space Requirements of Specialized Routing Schemes

An important feature of the specialized routing schemes, presented in [5]–[10], is that their forwarding algorithm does not have any impact on the memory space complexity as nodes only require information from the labels of their adjacent nodes to take the forwarding decision. Consequently, the number of entries in a routing table is equal to the node degree (Δ) and each entry stores at least the label of an adjacent node and the port that connects to it. Since, in the analyzed topologies, the value of Δ is sub-linear with respect to n (see Table I), then succinct node labels result in succinct routing tables. Therefore, the memory space complexity of the specialized schemes is $O(\Delta l)$ bits, where l denotes the size of a node label.

Theorem 1: The size of a node label defined by the BRP is $O(\log(n))$ bits, where n is the number of nodes in a BCube topology with the configuration parameters p and q.

Proof: The BRP assigns to each node a label denoted by an array $[a_q, a_{q-1}, \ldots, a_0]$, where $a_i \in [0, p-1]$ (see [5, Sec. 3.1]). Then, the size of a node label is $O(q \log(p))$ bits. Since $q < p$ and $p = \Delta$ (see Table I), the size of a label in terms of n is given by $O(\log(n^{1/4}))$ bits which can be expressed as $O(\log(n))$ bits. ■

Theorem 2: The size of a node label defined by the DFR is $O(\log(n))$ bits, where n is the number of nodes in a DCell topology with the configuration parameters p and q.

Proof: The DFR assigns to each node a label denoted by a tuple $(a_q, a_{q-1}, \ldots, a_0)$, where a_i indicates in which block

of level i the node is located (see [6, Sec. 3.2]). Since the level i connects $i+p$ blocks, then $a_i \in [0, i+p]$, where $0 \leq i < q$. Therefore, the size of a node label is $O(q \log(q+p))$ bits, where $q \leq 3$ and $p = \Delta$ (see Table I), and so the size of a label in terms of n is given by $O(\log(n^{1/6}))$ bits which can be expressed as $O(\log(n))$ bits. ■

Theorem 3: The size of a node label defined by the routing schemes based on permutation-sort [8]–[10] working on the following families of CG: Hypercube, Bubble-sort, Star, Transposition and Butterfly graphs is $O(\log(n)\log(\log(n)))$ bits, where n is the number of nodes in the CG that is defined by the parameter s.

Proof: The set of node labels used by the routing schemes based on permutation-sort is given by the set of permutations of the array $[0, \ldots, s-1]$. Then, the size of a node label is $O(s \log(s))$ bits. For Hypercube and Butterfly graphs, we have that $n \leq 2^s$ (see Table I), then $s \leq \log(n)$. For Bubble-sort, Star and Transposition graphs $n = s!$. If we apply Stirling's approximation, we have $s \leq c \log(n)$, where c is a constant. Therefore, for the analyzed CG, the size of a label in terms of n is given by $O(\log(n)\log(\log(n)))$ bits. ■

Theorems 1, 2 and 3 prove that the BRP, the DFR and the routing schemes based on permutation-sort provide sub-linear node labels. In addition, the two-level routing scheme working on Fat-tree topologies uses IPv4 addresses (that have constant size). Table II summarizes the memory space requirements of node labels used by the specialized routing schemes. From these results, Table III presents the memory space complexity of the specialized schemes, which (as explained above) is $O(\Delta l)$ bits. Tables II and III show that the specialized schemes provide sub-linear representation of both node labels and routing tables. This was to be expected since the schemes mentioned were designed to work on specific topologies in order to take advantage of the topological properties, such as sub-linear Δ and D.

C. Memory Space Requirements of the WMGR Scheme, Using the Word-Metric Space of the Free Group

In the WMGR scheme, the forwarding algorithm uses a Finite State Automaton (FSA) called *Diff* to compute the next node in the path. This automaton is related to the Word-Metric Space (WMS) used in the embedding process and its memory space complexity is $O(|\sigma_{Diff}|)$ bits, where σ_{Diff} is its set of states. The smallest automaton *Diff* is the automaton related to WMS the FG whose memory space complexity is $O(\Delta)$ bits (see [31, Example 1.1]) which does not impact the total memory space complexity of the WMGR scheme. In addition to the automaton *Diff*, nodes require information from the labels of their adjacent nodes to take the forwarding decision. Therefore, the memory space complexity of the WMGR scheme with embedding into the WMS of the FG is $O(\Delta l)$ bits. Since the WMGR scheme works in all the analyzed topologies, the memory space requirements of its node label, i.e. $O(l)$, depends on the network topology, specifically it depends on the values of D and Δ.

Theorem 4: The size of a node label defined by the WMGR scheme is $O(D \log(\Delta))$ bits, where D and Δ are the diameter of the network and node degree, respectively.

TABLE II
MEMORY SPACE COMPLEXITY OF NODE LABEL DEFINED BY SPECIALIZED ROUTING SCHEMES AND THE WMGR SCHEME

Topology	Specialized routing schemes	The WMGR scheme
BCube	$O(\log(n))$	$O(n^{1/4}\log(n))$
DCell		$O(\log(n))$
Fat-tree	$O(1)$	
Jellyfish	There is no specialized routing scheme	$O(\log(n)^2)$
Xpander		
Slim fly		$O(\log(n))$
Hypercube	$O(\log(n)\log(\log(n)))$	$O(\log(n)\log(\log(n)))$
Bubble-sort		$O(\log(n)^2\log(\log(n)))$
Star		$O(\log(n)\log(\log(n)))$
Transposition		
Butterfly		$O(\log(n))$

TABLE III
MEMORY SPACE COMPLEXITY OF SPECIALIZED ROUTING SCHEMES AND THE WMGR WITH EMBEDDING INTO THE WORD-METRIC OF THE FREE GROUP

Topology	Specialized routing schemes	The WMGR scheme
BCube	$O(n^{1/4}\log(n))$	$O(n^{1/2}\log(n))$
DCell	$O(n^{1/6}\log(n))$	
Fat-tree	$O(n^{1/3})$	$O(n^{1/3}\log(n))$
Jellyfish	There is no specialized routing scheme	$O(n^{1/c}\log(n)^2)$
Xpander		
Slim fly		$O(n^{1/2}\log(n))$
Hypercube	$O(\log(n)^2\log(\log(n)))$	$O(\log(n)^2\log(\log(n)))$
Bubble-sort		$O(\log(n)^3\log(\log(n)))$
Star		$O(\log(n)^2\log(\log(n)))$
Transposition	$O(\log(n)^3\log(\log(n)))$	$O(\log(n)^3\log(\log(n)))$
Butterfly	$O(\log(n)\log(\log(n)))$	$O(\log(n))$

Proof: In the WMGR scheme, a node label is given by a word over an alphabet A, where $|A| = \Delta$ (see Section IV-A). Each letter in a node label represents an edge and a whole node label represents one of the shortest paths in the network (see Section III-B). As length of the longest shortest path is D, then the size of a node label is $O(D \log(\Delta))$ bits. ■

Table II presents a comparison between the size of node labels used by the specialized routing schemes and the node labels used by the WMGR scheme. Note that the size of node labels used in the DCell, Hypercube, Star and Transposition graphs is the same in both kinds of schemes, i.e. the

specialized routing schemes and the WMGR scheme. On the other hand, the WMGR scheme achieves smaller node labels in the Butterfly graph than the node labels defined by the routing scheme based on permutation-sort. In BCube and the Bubble-sort graph, the node labels used by the WMGR scheme are larger than those used by their specialized schemes. In the Jellyfish, Xpander and Slim fly topologies, which do not have a specialized routing scheme, the WMGR provides poly-logarithmic node labels. Since the values of D and Δ are sub-linear in the analyzed topologies, then the WMGR scheme provides succinct node labels in all of them. From these results, Table III shows that the WMGR scheme with embedding into the WMS of the FG also has succinct memory space requirements.

D. Memory Space Requirements of the WMGR Scheme, Using the Word-Metric Space of the Underlying Cayley Graph

Since the smallest automaton *Diff* is the automaton related to the FG, then the network embedding into this WMS of the FG is the best embedding in terms of memory space complexity. As a result of this network embedding, the WMGR scheme achieves stretch 1 only in topologies based on trees (like the Fat-tree), whereas in the rest of topologies the stretch is bounded by D. In CG or topologies based on CG (like the BCube), the WMGR scheme is able to compute the shortest paths if the network is embedded into the WMS of its underlying CG (see Section V-F). The cost of achieving stretch 1 is an increase in the size of the automaton *Diff*. In this case, the memory space complexity of the WMGR scheme is $O(max\{|\sigma_{Diff}|, RT\})$ bits, where RT is the size of the routing table that is bounded by $O(\Delta D \log \Delta)$ bits (see the previous subsection).

From Section IV-B3, we know that $|\sigma_{Diff}|$ is equal to the number of nodes in the subgraph $B(Id, k)$ of the underlying CG, i.e. $|\sigma_{Diff}| = |V(B(Id, k))|$. This subgraph is given by the graph with radius k around to the identity node. In [3], it is shown that the value of k is bounded by the hyperbolicity (δ) of the CG, according to the relation $\delta > \frac{k}{2}$, where $\delta \leq \frac{D}{2}$, then $k \leq D$. Note that if $k = D$, then $B(Id, k)$ represents the whole network graph and $|\sigma_{Diff}| = n$, which, in terms of memory space complexity, is equivalent to having full routing tables. Nevertheless, in [33], when the value of k was computed through computer simulation for the evaluated CG, we found that the value of k is constant in the Hypercube, Transposition and Bubble-sort graphs, whereas in the Star and Butterfly graphs, k grows proportionally in terms of D. Actually in the Butterfly graphs the value of k is very closed to D.

Fig. 7 shows the ratio between $|\sigma_{Diff}|$ and n for the automaton related to the Hypercube, Bubble-sort, Star, Transposition and Butterfly graphs. As we can see, the WMGR scheme has the lowest memory space complexity when working on the Hypercube, Bubble-sort and Transposition graphs, where $|\sigma_{Diff}|$ is less that 10% of n, for $n > 720$. These results are consistent with the fact that, as mentioned above, such CG have a constant value of k. In contrast, in the Star and Butterfly graphs (where the value of k is proportional to D), $|\sigma_{Diff}|$ is less that the 55% of n, for $n \geq 64$.

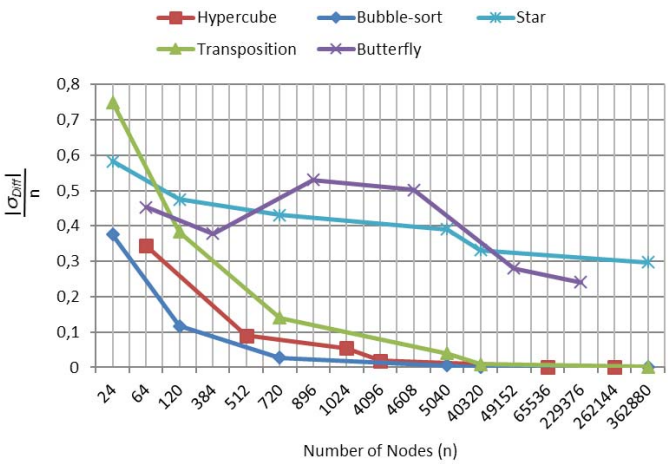

Fig. 7. Ratio between the number of states of the *Diff* automaton ($|\sigma_{Diff}|$) and the number of nodes in the network (n).

The results shown in Fig. 7 were obtained by using the software package *Knuth-Bendix on Monoids and Automatic Groups* (KBMAG) [36], which provides an implementation of the Knuth-Bendix procedure. The group presentation of a CG provides the input in this procedure, and a lexicographic order over the generating set (this order is also used in the embedding process, see Section IV-A). Then, the procedure returns a *ShortLex* Automatic Structure (SAS) that contains the automaton *Diff*.

A note of caution is required here as for some automatic groups it is not possible to compute an SAS for any lexicographic order. Moreover, different lexicographic orders may result in SAS of different sizes (see [35, Sec. 13.2.1]). It is unknown when a group is *ShortLex* automatic with respect to some lexicographic order and which lexicographic order leads to the smallest SAS. For instance, Coxeter groups such as the group presentation related to the Bubble-sort graph, are *ShortLex* automatic only with some lexicographic orders. On the other hand, *word hyperbolic* groups [37], such as the FG, can be encoded in an SAS, no matter what lexicographic order has been chosen. The results presented in Fig. 7 were obtained by using the lexicographic order given by the definitions of the generating sets of CG introduced in [38].

E. Forwarding Decision Time

In the complexity analysis of the forwarding decision time we do not consider the time required to search in the routing table, and we only take into account the operations that have non-constant execution time.

Theorem 5: The forwarding decision time in the BRP and the DFR takes $O(\log(n))$ time units, where n is the number of nodes in the network.

Proof: Given a source and a destination node, the BRP and the DFR compute the next node in the path by comparing the label of the source node with the label of the destination node (see [5, Sec. 3.2], [6]). Assuming that comparing two node labels takes an amount of time proportional to the label size, which is $O(\log(n))$ bits (see Theorems 1 and 2), then the forwarding decision time in the BRP and the DFR takes $O(\log(n))$ time units. ■

TABLE IV
FORWARDING DECISION TIME COMPLEXITY OF SPECIALIZED ROUTING SCHEMES AND THE WMGR SCHEME

Topology	Specialized routing schemes	The WMGR scheme (free group)	The WMGR scheme (underlying Cayley graph)
Fat-tree[a]	$O(1)$	$O(\log(n)^2)$	
BCube	$O(\log(n))$	$O(n^{5/16}\log(n)^2)$	$O(n^{1/16}\log(n)^2)$
DCell		$O(n^{1/6}\log(n)^2)$	There is no underlying Cayley graph
Jellyfish	There is no specialized routing scheme	$O(n^{1/c}\log(n)^4)$	
Xpander			
Slim fly		$O(n^{1/2}\log(n)^2)$	
Hypercube	$O(\log(n)^2\log(\log(n)))$	$O(\log(n)^3\log(\log(n))^2)$	$O(\log(n)^2\log(\log(n))^2)$
Bubble-sort		$O(\log(n)^5\log(\log(n))^2)$	$O(\log(n)^4\log(\log(n))^2)$
Star		$O(\log(n)^3\log(\log(n))^2)$	$O(\log(n)^2\log(\log(n))^2)$
Transposition	$O(\log(n)^3\log(\log(n)))$	$O(\log(n)^4\log(\log(n))^2)$	
Butterfly	$O(\log(n)\log(\log(n)))$	$O(\log(n)^2)$	$O(\log(n)^2)$

[a]The underlying Cayley graph of Fat-tree is the Cayley graph of free group.

Theorem 6: The forwarding decision time in the routing schemes that are based on permutation-sort takes $O(\Delta\log(n)\log(\log(n)))$ time units, where Δ is the maximum node degree and n is the number of nodes in the CG.

Proof: Given a source and a destination node, the routing schemes based on permutation-sort compute the next node in the path by comparing the label of the adjacent nodes (to the source node) with the label of the destination node. Then, the source node executes Δ comparisons. Assuming that compering two node labels takes an amount total of time proportional to the label size, which is $O(\log(n)\log(\log(n)))$ bits (see Theorem 3), then the forwarding decision takes, at most $O(\Delta\log(n)\log(\log(n)))$ time units. ■

Theorem 7: The forwarding decision time in the WMGR scheme with embedding into the WMS of the FG takes $O(\Delta D^2\log(\Delta)^2)$ time units, where Δ and D are the maximum node degree and the diameter, respectively.

Proof: The WMGR scheme with embedding into the WMS of the FG computes the next node in the path by solving the Minimum Word Problem (MWP) Δ times (see Algorithm 2). The WMGR scheme uses an FSA called *Diff* to solve MWP in $O(|u^{-1}u|^2)$ time units, where u and v are node labels (see [27, Sec. 2.5]). From Table II, the size of a node label is $(D\log(\Delta))$ bits, then the forwarding decision time in the WMGR scheme with embedding into the WMS of the FG takes $O(\Delta D^2\log(\Delta)^2)$ time units. ■

Theorem 8: The forwarding decision time in the WMGR scheme with embedding into the WMS of the underlying CG takes $O(D^2\log(\Delta)^2)$ time units, where Δ and D are the maximum node degree and the diameter, respectively.

Proof: The WMGR scheme with embedding into the WMS of the underlying CG computes the next node in the path by solving the MWP once (see Algorithm 3). From Theorem 7, the WMGR scheme solves the MWP in $O(D^2\log(\Delta)^2)$ time units. ■

Theorems 5, 6, 7 and 8 prove that the forwarding decisions in the analyzed routing schemes are taken in time proportional to the size of node labels. Then, in the BRP, the DFR, the routing schemes based on permutation-sort and the WMGR scheme, succinct node labels result in fast forwarding decisions. On the other hand, the two-level routing scheme working on Fat-tree topologies takes forwarding decisions in a constant time because it only perform searches in routing tables (see [7, Sec. 3.5]).

Table IV compares the complexity of forwarding decision time for the specialized schemes and the WMGR scheme with both network embeddings: 1) into the WMS of the FG and 2) into the WMS of the underlying CG. BCube has an underlying Hypercube and topologies based on trees (like Fat-tree) are related to the WMS of the FG due to the CG of the FG also being a tree. Conversely, DCell, Jellyfish, Xpander and Slim fly topologies do not have an underlying CG. Table IV shows that the forwarding decision in the WMGR scheme is faster when the network is embedded into the WMS of its underlying CG than when it is embedded into the WMS of the FG. This is because the forwarding algorithm must solve the MWP only once instead of Δ times (see Section IV-C). In spite of the WMGR scheme spending more time to take forwarding decisions than the specialized routing schemes, both kinds of schemes have sub-linear forwarding decision time complexity.

F. Stretch

The WMGR scheme computes the shortest paths when the network is embedded into its underlying CG. However, this

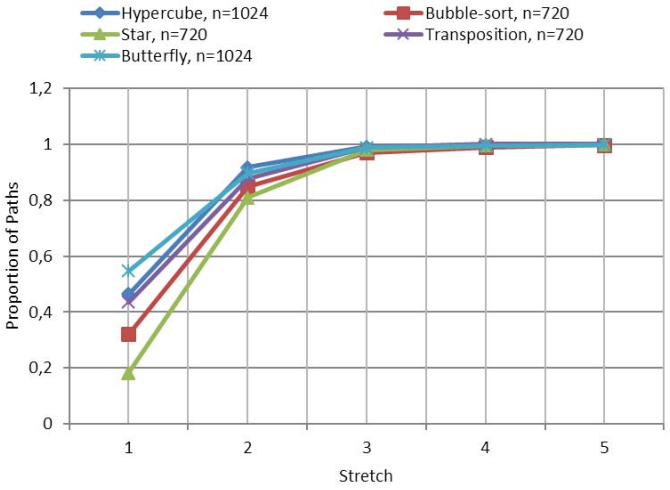

Fig. 8. Cumulative distributed function of stretch for the WMGR scheme (free group) working on five families of Cayley graphs of different sizes.

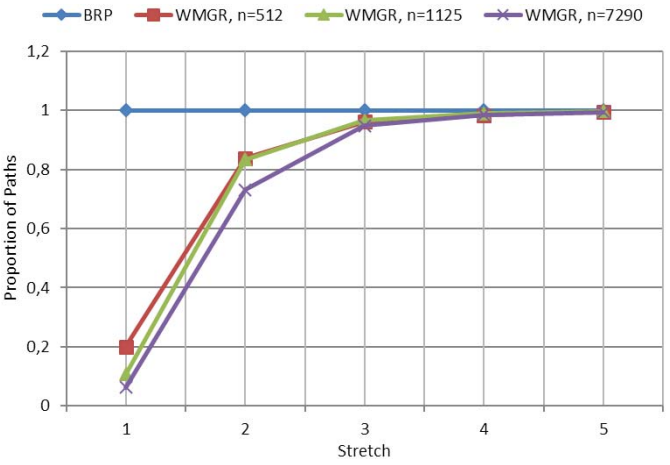

Fig. 9. Cumulative distributed function of stretch for the BRP and the WMGR scheme (free group) working on three BCube topologies of different sizes.

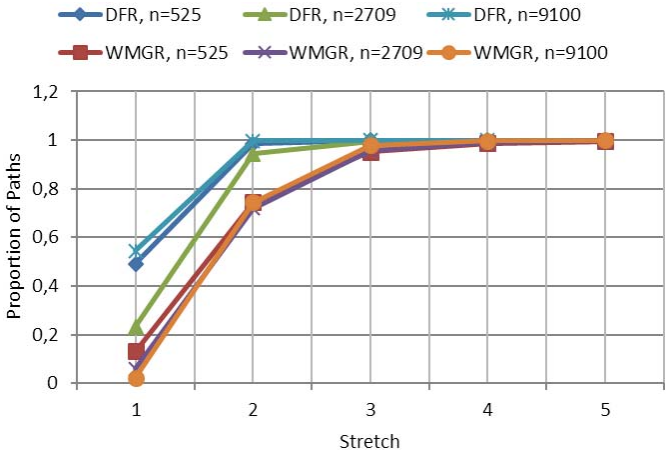

Fig. 10. Cumulative distributed function of stretch for the DFR and the WMGR scheme (free group) working on three DCell topologies of different sizes.

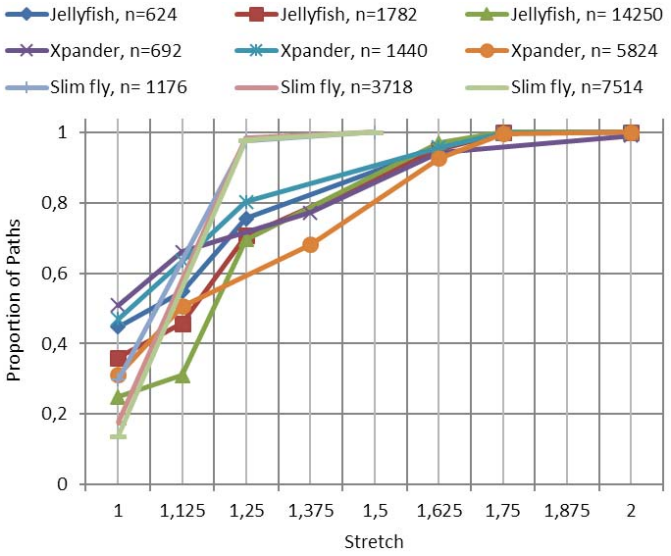

Fig. 11. Cumulative distributed function of stretch for the WMGR scheme (free group) working on Jellyfish, Xpander and Slim fly topologies of different sizes.

embedding increases the memory space complexity of the scheme. If the priority is to save memory space, the network can be embedded into the WMS of the FG (see Section V-C), although in this case the resulting stretch is bounded by D.

Theorem 9: Let Γ be a connected and finite graph representing a network topology whose diameter is denoted by D. If Γ is embedded into the WMS of the FG, then the stretch of the WMGR scheme is bounded by D.

Proof: Let u and v be nodes in $V(\Gamma)$. If the distance in the metric graph between u and v is at least two (i.e. $d_\Gamma(u,v) \leq 2$), then, in the worst case, the WMGR scheme routes data packets along the underlying Breadth First Search (BFS) tree T_Γ (see Section IV-A). Therefore, the stretch of the WMGR scheme is given by $d_{T_\Gamma}(u,v)/d_\Gamma(u,v)$, where $d_{T_\Gamma}(u,v) \leq 2D$, because the diameter of BFS tree T_Γ is at most $2D$. The resulting stretch, in the worst case, is $2D/2 = D$. ∎

From Theorem 9, when the network is embedded into the WMS of the FG, the WMGR scheme has stretch bounded by D, which is poly-logarithmic in the BCube, Hypercube, Bubble-sort, Star, Transposition, Butterfly, Jellyfish and Xpander topologies. Meanwhile, in the DCell, Fat-tree and Slim fly topologies, the stretch is constant (see Table I). However, the stretch was evaluated through computer simulation and the results obtained show that in all of the evaluated topologies, at least 75% of the pairs of nodes computed by the WMGR scheme with embedding into the WM of the FG have stretch less than 2 (see Figs. 8, 9, 10 and 11). Fig. 8 shows the Cumulative Distribution Function (CDF) of stretch for the WMGR scheme working on five families of CG. As we can see, in all cases at least 85% of the computed paths have stretch less than 2.

Figures 9 and 10 present a comparison between the CDF of stretch for the WMGR scheme and the CDF of stretch for the BRP and the DFR, respectively. In Fig. 9, BRP is a shortest path scheme. In the case of the WMGR scheme, only a small number (0.6%) of the computed paths reach stretch higher than 5. In fact, 83% of the computed paths have stretch of less

than 2. Similarly, Fig. 10 shows that 0.5% of paths computed by the DFR and the WMGR scheme have stretch higher than 5. Seventy-eight percent of paths found by the WMGR scheme have stretch less than 2. Finally, Fig. 11 shows that in the Jellyfish, Xpander and Slim fly topologies, 99% of the paths computed by the WMGR scheme have stretch less than 1.75.

Theorem 10: Let $\Gamma(G, S)$ be a graph representing an underlying CG of a network topology. If $\Gamma(G, S)$ is embedded into its own WMS, the WMGR scheme computes the shortest paths in Γ.

Proof: The WMGR scheme performs an embedding denoted by (7), where network nodes are given by the elements of G. Following the Definition 11, π is an isometric embedding and therefore the WMGR scheme computes the shortest paths in the network with underlying $\Gamma(G, S)$. ■

VI. Conclusion

In this paper, we have presented the Word-Metric-based Greedy Routing (WMGR) scheme for Data Center Networks (DCN). Unlike other routing schemes for DCN, which are either topology dependent (specialized) or are not compact, our scheme is topology independent (generic) and compact in DCN. The scheme is based on an embedding process, where nodes are assigned to coordinates (or labels) in the Word-Metric Space (WMS) of an algebraic group, and on a greedy forwarding strategy, where nodes forward packets to the closest neighbor to the destination in this WMS. The forwarding algorithm is implemented using a finite state automaton that encodes information related to the network topology and that can be stored in an efficient way.

We have evaluated our scheme and compared it to other routing schemes in different topologies: the specialized routing schemes BCube Routing Protocol (BRP), DCell Fault-tolerant Routing protocol (DFR), two-level routing scheme and the routing schemes based on permutation-sort, and the topologies BCube, DCell, Fat-tree, Jellyfish, Xpander, Slim fly, and five Cayley Graphs (CG), Hypercube, Bubble-sort, Star, Transposition and Butterfly. In all topologies we have proved that our scheme is scalable (sub-linear growth with respect to the number of nodes) in memory space requirements (node label, routing table and forwarding algorithm) and forwarding decision time. Concerning stretch, our scheme is able to compute the shortest paths (stretch 1) in topologies based on CG and also on trees (e.g, Fat-tree), while in the rest of topologies it achieves a stretch that grows logarithmically (with respect to the number of nodes). Moreover, simulation results have shown that many of the paths remains far below this upper bound (at least 75% of them have stretch less than 2).

We have also compared the performance of our scheme in CG topologies using the WMS of two different algebraic groups. With the WMS of the underlying CG, the scheme achieves stretch 1 at the cost of an increase in memory space requirements, while with the WMS of the Free Group (FG), the memory space requirements decrease but the stretch is higher; a behavior that shows the well-known trade-off between stretch and memory space requirements in compact routing schemes [1].

We have made a complexity analysis for the memory space requirements, forwarding decision time and stretch of several specialized routing schemes. These schemes exploit the sub-lineal values of diameter and node degree to become compact in a particular DCN topology. Our scheme also exploits the same topological properties but it is compact in all the analysed topologies. Note that our scheme would provide a scalable solution in any topology that meet these properties.

References

[1] D. Peleg and E. Upfal, "A tradeoff between space and efficiency for routing tables," in *Proc. 20th Annu. ACM Symp. Theory Comput. (STOC)*, New York, NY, USA, 1988, pp. 43–52. [Online]. Available: http://doi.acm.org/10.1145/62212.62217

[2] M. Thorup and U. Zwick, "Compact routing schemes," in *Proc. 20th Annu. ACM Symp. Parallel Algorithms Archit. (SPAA)*, New York, NY, USA, 2001, pp. 1–10. [Online]. Available: http://doi.acm.org/10.1145/378580.378581

[3] D. Coudert and G. Ducoffe, "Data center interconnection networks are not hyperbolic," *Theor. Comput. Sci.*, vol. 639, pp. 72–90, Aug. 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S030439751630161X

[4] Y. Liu, J. K. Muppala, M. Veeraraghavan, D. Lin, and M. Hamdi, "Data center network topologies: Research proposals," in *Data Center Networks* (SpringerBriefs in Computer Science). Cham, Switzerland: Springer, 2013.

[5] C. Guo *et al.*, "BCube: A high performance, server-centric network architecture for modular data centers," in *Proc. ACM SIGCOMM Conf. Data Commun. (SIGCOMM)*, New York, NY, USA, 2009, pp. 63–74.

[6] C. Guo *et al.*, "DCell: A scalable and fault-tolerant network structure for data centers," in *Proc. ACM SIGCOMM Conf. Data Commun. (SIGCOMM)*, New York, NY, USA, 2008, pp. 75–86. [Online]. Available: http://doi.acm.org/10.1145/1402958.1402968

[7] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, 2008. [Online]. Available: http://doi.acm.org/10.1145/1402946.1402967

[8] G. D. Stamoulis and J. N. Tsitsiklis, "The efficiency of greedy routing in hypercubes and butterflies," in *Proc. 3rd Annu. ACM Symp. Parallel Algorithms Archit. (SPAA)*, New York, NY, USA, 1991, pp. 248–259. [Online]. Available: http://doi.acm.org/10.1145/113379.113402

[9] S. B. Akers, B. Krishnamurthy, and D. Harel, "The star graph: An attractive alternative to the n-cube," in *Proc. Int. Conf. Parallel Process.*, 1987, pp. 393–400.

[10] S. B. Akers and B. Krishnamurthy, "A group-theoretic model for symmetric interconnection networks," *IEEE Trans. Comput.*, vol. 38, no. 4, pp. 555–566, Apr. 1989.

[11] J. Y. Yen, "Finding the K shortest loopless paths in a network," *Manage. Sci.*, vol. 17, no. 11, pp. 712–716, 1971.

[12] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, "Jellyfish: Networking data centers, randomly," in *Proc. 3rd USENIX Conf. Hot Topics Cloud Comput. (HotCloud)*, Berkeley, CA, USA, 2011, p. 12. [Online]. Available: http://dl.acm.org/citation.cfm?id=2170444.2170456

[13] A. Valadarsky, M. Dinitz, and M. Schapira, "Xpander: Unveiling the secrets of high-performance datacenters," in *Proc. 14th ACM Workshop Hot Topics Netw. (HotNets-XIV)*, New York, NY, USA, 2015, pp. 16:1–16:7. [Online]. Available: http://doi.acm.org/10.1145/2834050.2834059

[14] M. Besta and T. Hoefler, "Slim fly: A cost effective low-diameter network topology," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal. (SC)*, Piscataway, NJ, USA, Nov. 2014, pp. 348–359. [Online]. Available: https://doi.org/10.1109/SC.2014.34

[15] M. Camelo, D. Papadimitriou, L. Fàbrega, and P. Vilà, "Geometric routing with word-metric spaces," *IEEE Commun. Lett.*, vol. 18, no. 12, pp. 2125–2128, Dec. 2014.

[16] S. T. Schibell and R. M. Stafford, "Processor interconnection networks from cayley graphs," *Discrete Appl. Math.*, vol. 40, no. 3, pp. 333–357, 1992. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0166218X9290005U

[17] K. W. Tang and B. W. Arden, "Vertex-transitivity and routing for Cayley graphs in GCR representations," in *Proc. ACM/SIGAPP Symp. Appl. Comput., Technol. Challenges (SAC)*, New York, NY, USA, 1992, pp. 1180–1187.

[18] M. Miller and J. Širáň, "Moore graphs and beyond: A survey of the degree/diameter problem," *Electron. J. Combinat.*, vol. 20, no. 2, May 2013, Art. no. DS14.
[19] C. H. Papadimitriou and D. Ratajczak, "On a conjecture related to geometric routing," *Theor. Comput. Sci.*, vol. 344, no. 1, pp. 3–14, Nov. 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0304397505003725
[20] R. Dhandapani, "Greedy drawings of triangulations," in *Proc. 19th Annu. ACM-SIAM Symp. Discrete Algorithms (SODA)*, Philadelphia, PA, USA, 2008, pp. 102–111. [Online]. Available: http://dl.acm.org/citation.cfm?id=1347082.1347094
[21] R. Kleinberg, "Geographic routing using hyperbolic space," in *Proc. 26th IEEE Int. Conf. Comput. Commun. (INFOCOM)*, May 2007, pp. 1902–1909.
[22] D. Eppstein and M. T. Goodrich, "Succinct greedy geometric routing using hyperbolic geometry," *IEEE Trans. Comput.*, vol. 60, no. 11, pp. 1571–1580, Nov. 2011.
[23] J. Newsome and D. Song, "GEM: Graph EMbedding for routing and data-centric storage in sensor networks without geographic information," in *Proc. 1st Int. Conf. Embedded Netw. Sensor Syst. (SenSys)*, New York, NY, USA, 2003, pp. 76–88.
[24] M. R. Bridson and A. Haefliger, "Metric spaces of non-positive curvature," in *Grundlehren der Mathematischen Wissenschaften*. Berlin, Germany: Springer, 1999. [Online]. Available: http://opac.inria.fr/record=b1095970
[25] D. J. S. Robinson, "A course in the theory of groups," in *Graduate Texts in Mathematics*, vol. 80. Berlin, Germany: Springer-Verlag, 1996.
[26] C. H. Papadimitriou and D. Ratajczak, "On a conjecture related to geometric routing," in *Proc. ALGOSENSORS*, 2004, pp. 9–17.
[27] D. B. A. Epstein, *Word Processing in Groups*. Natick, MA, USA: A. K. Peters, 1992.
[28] J. Meier, *Groups, Graphs Trees*. Cambridge, U.K.: Cambridge Univ. Press, 2008. [Online]. Available: http://dx.doi.org/10.1017/CBO9781139167505
[29] S. Even and O. Goldreich, "The minimum-length generator sequence problem is NP-hard," *J. Algorithms*, vol. 2, no. 3, pp. 311–313, 1981.
[30] C. Boulinier, A. K. Datta, L. L. Larmore, and F. Petit, "Space efficient and time optimal distributed BFS tree construction," *Inf. Process. Lett.*, vol. 108, no. 5, pp. 273–278, Nov. 2008. [Online]. Available: http://dx.doi.org/10.1016/j.ipl.2008.05.016
[31] D. B. A. Epstein, A. R. Iano-Fletcher, and U. Zwick, "Growth functions and automatic groups," *Experiment. Math.*, vol. 5, no. 4, pp. 297–315, 1996. [Online]. Available: http://projecteuclid.org/euclid.em/1047565448
[32] D. Knuth and P. Bendix, "Simple word problems in Universal Algebras," in *Automation of Reasoning* (Symbolic Computation), J. Siekmann and G. Wrightson, Eds. Berlin, Germany: Springer, 1983, pp. 342–376.
[33] M. Camelo, P. Vilà, L. Fàbrega, and D. Papadimitriou, "Cayley-graph-based data centers and space requirements of a routing scheme using automata," in *Proc. IEEE 34th Int. Conf. Distrib. Comput. Syst. Workshops (ICDCSW)*, Jun. 2014, pp. 63–69.
[34] M. Camelo, D. Papadimitriou, L. Fàbrega, and P. Vilà, "Efficient routing in data center with underlying cayley graph," in *Proc. 5th Workshop Complex Netw. CompleNet*, 2014, pp. 189–197.
[35] D. F. Holt, B. Eick, and E. A. O'Brien, *Handbook of Computational Group Theory* (Discrete Mathematics and its Applications). Boca Raton, FL, USA: Chapman & Hall, 2005. [Online]. Available: http://opac.inria.fr/record=b1102239
[36] D. Holt. (2009). *KBMAG Package: A Knuth-Bendix on Monoids, and Automatic Groups*. Accessed: Jan. 15, 2015. [Online]. Available: http://homepages.warwick.ac.uk/mareg/kbmag/
[37] M. Gromov, "Hyperbolic groups," in *Essays in Group Theory* (Mathematical Sciences Research Institute Publications), vol. 8. New York, NY, USA: Springer, 1987, pp. 263–275.
[38] M.-C. Heydemann and B. Ducourthial, "Cayley graphs and interconnection networks," Univ. Paris-Sud, Orsay, France, Tech. Rep. RR 1122, 1997. [Online]. Available: http://opac.inria.fr/record=b1038371

Daniela Aguirre-Guerrero received the bachelor's degree in telematics engineering from the National Polytechnic Institute, Mexico, in 2010, and the master's degree in information technology from Autonomous Metropolitan University, Mexico, in 2014. She is currently pursuing the Ph.D. degree with the University of Girona, Catalonia, Spain. Her research interests include computation in large graphs, complex networks, and graph theory.

Miguel Camelo received the bachelor's degree in electronic engineering from the University of Ibague, Colombia, in 2006, the master's degree in systems and computer engineering from the University of Los Andes, Colombia, in 2010, and the Ph.D. degree in computer engineering from the University of Girona, Spain, in 2014. He is currently a Researcher with the Department of Mathematics and Computer Sciences, University of Antwerp, Belgium. He has authored or co-authored several papers in journals and international conferences and he has involved in several Spanish, Belgian, and European research projects. His research interests are in the fields of control and management of communication networks and routing in complex networks using group theory, evolutionary algorithms, and machine learning.

Lluís Fàbrega received the bachelor's degree in telecommunications engineering and the master's degree in mobile communications from the Polytechnical University of Catalonia, in 1995 and 1996, respectively, and the Ph.D. degree in computer science from the University of Girona (UdG) in 2008. He has been an Associate Professor in computer science with UdG since 2008. He has involved in several Spanish and EU (Celtic, FP7) research projects. He has co-authored several papers in journals and international conferences. His research interests include the design and performance evaluation of routing, traffic engineering, and quality of service mechanisms in the Internet and in connection oriented network technologies.

Pere Vilà received the degree in computer science engineering from the Polytechnic University of Catalonia in 1997 and the Ph.D. degree in computer science from the University of Girona in 2004. He is currently an Associate Professor in computer science with the University of Girona. He has authored or co-authored around 50 papers in his areas of interest. His research interests include new routing algorithms for future Internet, network protection and robustness, complex networks, network management, and control. He served as a member for program committee in several international conferences and an Associate Editor for the *International Journal of Communication Systems*.